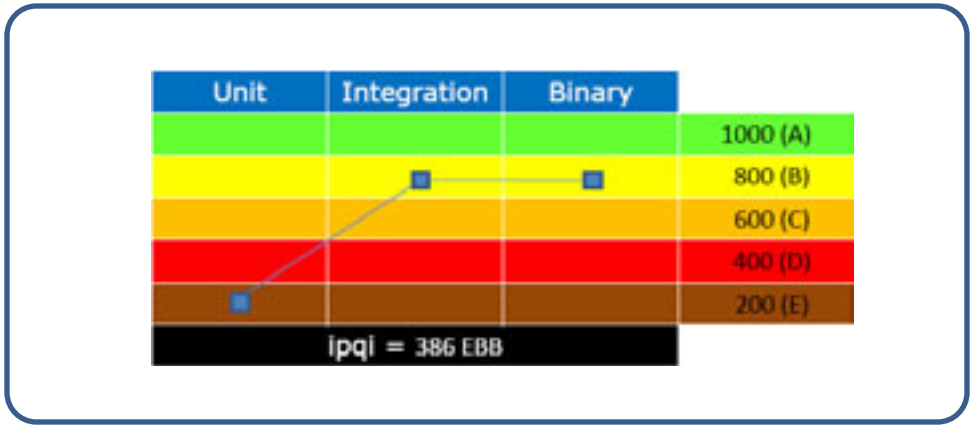
Software Quality Certification Using the “IPQI” Model



Outline

- Introduction

What is software product quality and how to model it

- **IPQI** model

model description/reasoning/hypotheses

- Preliminary findings

Product quality

Internal
(relevant for the project)

"How well does the product support my development tasks"

KPI: Complexity

External
(relevant for the product users)

"How well does the product work"

KPI: Test coverage
KPI: Defect density
KPI: Running costs

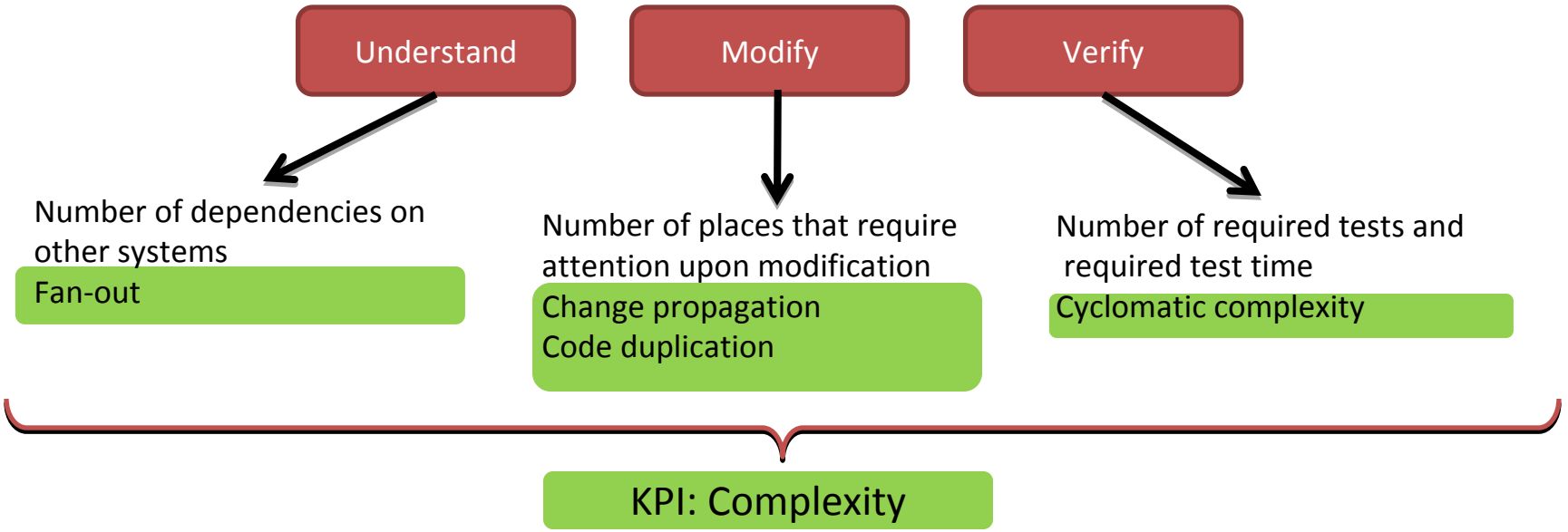
SolidSource

Internal Quality

SIG TÜViT product certification model

ISO9126: Maintainability

How difficult is it to understand, modify and verify the product?



SolidSource

External Quality

ISO9126: Functionality + Reliability + Usability + Efficiency + Portability

KPI: Test coverage
KPI: Defect density

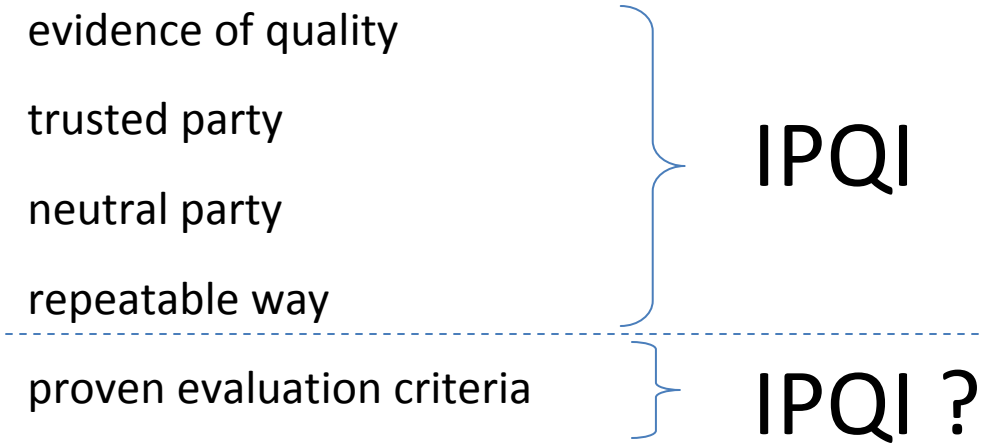
KPI: Running costs

What is the chance the product will fail to meet its expected behavior

IPQI

Certification:

evidence of quality given by a trusted neutral party that uses proven evaluation criteria in a repeatable way.



Definition:

IPQI (integrated Product Quality Index) is a software metric that expresses in a single normative figure the external quality of a software product related to functionality and reliability.

Target audience:

- **Software developers and testers**

Deliver high-quality software by assessing and improving quality during product development and maintenance.

- **Software manufacturers**

Certify the quality of a software product prior to releasing it to its intended customer(s) and/or end-user(s).

- **Potential buyers**

Assess the quality of end-user software products or third-party software stacks.

SolidSource

IPQI

Unit testing quality
The overall quality of the product components taken in isolation

Binary unit quality
The overall quality of third party components for which source code is not available

Integration testing quality
The overall quality of the integration of the product components

The IPQI Model: Unit Testing Quality (QU)

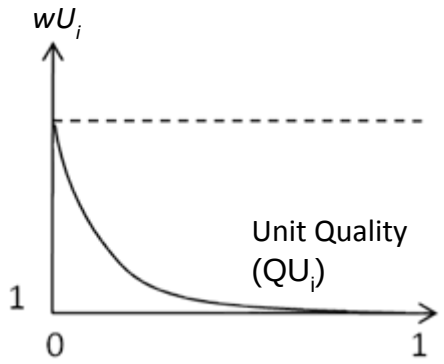
The unit testing quality (QU_i) associated with a source code unit (i) is based on a branch related code coverage measure ($MCDC_i$) that is biased with statement coverage (SC_i). This ensures that both branches and statements are sufficiently covered in order to obtain high quality unit testing.

$$QU_i = MCDC_i \cdot SC_i$$

To compute the overall quality of the unit testing aspect in the system one needs to consider the way the quality of a system is perceived. Namely, the perceived quality of a system is greatly influenced by its 'weakest link'.

To this end, the quality based weighting factor wU_i (determined experimentally) is used for aggregating the individual QU_i values of the units in one overall value.

$$QU = \frac{\sum_i wU_i \cdot QU_i}{\sum_i wU_i} \cdot 1000$$

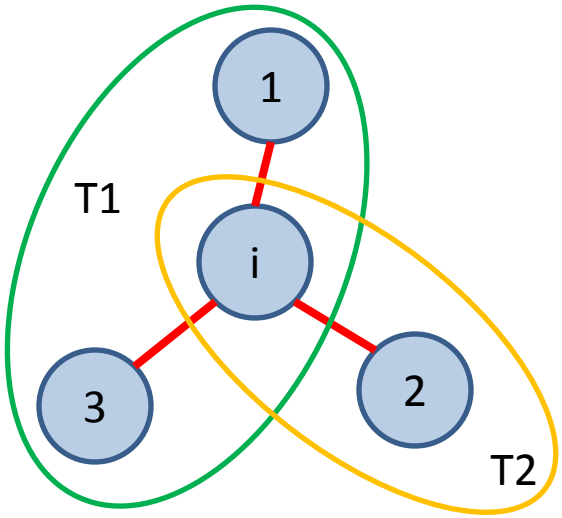


The IPQI Model: Integration Testing Quality (QI)

SolidSource

The integration testing value (QI_i) for one unit is computed as a weighted sum of all integrations of *i* with other units, as present in the final product.

$$QI_i = \frac{\sum_j wIU_j \cdot QU_j}{\sum_j wIU_j}$$



- Each integration QU_j can be measured using the unit coverage formula applied to test applications (as a whole) that should at least contain two units from the final product: *i* and *j*.
- One integration has to be considered for each pair.

The weighting factor wUI_i of each unit is quality based and should be determined experimentally.

The overall integration testing quality *QI* is computed as the weighted average of the integration quality values specific to each unit.

$$QI = \frac{\sum_i wI_i \cdot QI_i}{\sum_j wI_i}$$

The weighting factor wI_i of each unit is quality based and should be determined experimentally.

The IPQI Model: Binary Unit Quality (ipqi)

SolidSource

It is not possible/practical to reliably assess the code quality of a binary unit using a black box testing approach. Therefore, the quality of such a unit should be assessed by its developers using a white box testing approach at development time, and passed further together with the binary unit. The quality value should be assessed by developers using the integrated formula for ipqi presented next.

$$QB = \frac{\sum_i wB_i \cdot IPQI_i}{\sum_i wB_i}$$

The weighting factor wB_i of each binary unit is quality based and should be determined experimentally

Binary units do not have an integration factor in the overall ipqi formula. The integration aspects related to binary units are considered to be included in:

- The integration tests performed by the developers of the binary units;
- The integration tests on the source code units of the current system.

The IPQI Model: Integrated Formula

SolidSource

$$ipqi_{total} = \frac{\sum_i wO_i \cdot ipqi_i}{\sum_i wO_i}$$

$$ipqi_i = \begin{cases} \sqrt{QU_i \cdot QI_i} \cdot 1000 | \ source\ code\ units \\ reported\ ipqi | \ binary\ objects \end{cases}$$

wO_i an experimentally set weighting factor based on $ipqi_i$

Letter	Interval	Certification level	Typical Industry Segment
A	801 – 1000	Excellent	Aerospace, Medical
B	601 – 800	Good	Telecom
C	401 – 600	Fair	Finance, Automotive
D	201 – 400	Poor	Entertainment
E	0 – 200	Very poor	-

Example: *ipqi = 750 ABC* indicates:

- The software product exists of source code units of excellent quality (A)
- These units are well integrated (B)
- The binary units have fair quality (C)

The IPQI Model: Metric properties

SolidSource

- **Simple**

Although complex metrics may be more accurate than simple ones, and most developers will be able to understand them (if they are willing to put the time into it), the popularity, effectiveness, and usefulness of most metrics (software or otherwise) is inversely proportional to their complexity.

- **Positive**

A metric is considered to be positive if one wants the quantity it measures to go up.

- **Controllable**

The success of an effort should be tied to one or more metrics that can be controlled. Each (combination of) metric(s) should enable one to define measures to positively influence the metric(s).

- **Automatable**

If calculating a metric requires manual effort it will quickly turn into a chore and it will not be tracked as frequently or as accurately as it should be. It is essential that whatever one decides to measure can be easily automated and will require little or no human effort to collect the data and calculate the result.

The IPQI Model: Example

SolidSource

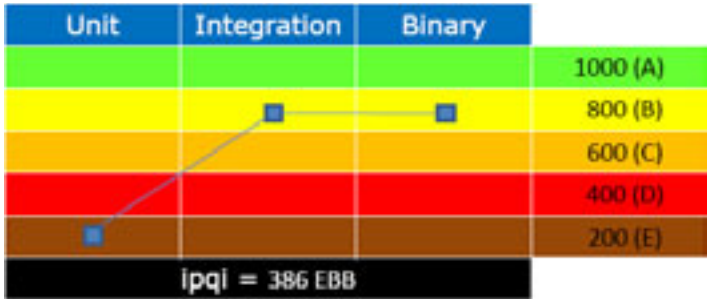
Unit name	Type	Unit Quality (QU_i)	Integration Quality (QI_i)	Reported Quality ($IPQI_i$)	Quality bias (wU)	Quality bias (wI)	Quality bias (wB)	Quality bias (wO)
A	Src	0.1	0.6		10	2		7
B	Src	0.2	0.9		8	1		4
C	Binary			700	1		1.5	1.5
D	Binary			800	1		1	1

*QU = (10/18 * 0.1 + 8/18 * 0.2)*1000 = 144.44 (E)*

*QI = (2*0.6/3 + 0.9/3) * 1000 = 700 (B)*

QB = 1.5/2.5 700 + 1/2.5 * 800 = 740 (B)*

*IPQI = 7/13.5*0.24*1000 +*
*4/13.5*0.42*1000 +*
*1.5/13.5*700 +*
*1/13.5*800 = 386*



The IPQI Model: Open issues

SolidSource

Unit granularity level: The definition of unit can be very different from one organization to another. In general IPQI favors the decomposition of a system in small units that are easy to test. In order to avoid abusing this decomposition to avoid the integration component of IPQI one needs to make sure during integration testing that interfaces are used by the right components and not by stubs or drivers.

Integration testing: To facilitate integration testing without losing assessment relevance, a data flow analysis approach can be used. This can identify all code that is relevant for making a function call across units. Consequently, only code in the identified area should be considered for test coverage assessment during integration testing. This would require additional technology for implementing the tool support.

Programming languages: The test coverage concepts used are quite generic but one needs to verify their applicability to (the combination of) any language.

Bias factors. The bias factors defined for modeling the influence of the 'weakest link' on the overall quality need to be set experimentally. To this end, a number of calibration studies are required.

Mathematical correctness: The IPQI model trades intuition and perception for mathematical elegance.

Preliminary findings

SolidSource

• Case studies are being performed to set bias factors and investigate usefulness in industry (i.e., proven evaluation criteria).

• Industry confirms the need for (external) product quality certification.

• There are few organizations that use code coverage in assessing the quality of the testing. In general, industry has limited experience in assessing product quality , yet it reports high focus and investment levels in process certification / improvement.

• Although testing frameworks are present they lack the necessary reporting capabilities. Available Information is incomplete and inconsistent. Using IPQI can require important modifications in the testing infrastructure and supporting technologies.

Conclusions

SolidSource

• IPQI can model the perceived quality of a product in one figure, and therefore is very suitable for making business decisions.

• IPQI calculation can be completely automated and therefore very suitable and useful for supporting a product quality certification model.

• It builds on basic blocks for software quality assessment (test code coverage) and therefore is relatively easy to introduce in practice.

• It requires access to advanced technologies for code analysis and appropriate reporting (code coverage monitoring, data flow analysis) and therefore is relatively difficult to introduce in practice.

Thank you!

S.L. Voinea

Luchthavenweg 144A

5657 EA Eindhoven

Netherlands

Tel: 0614.36.38.42

E-mail:
info@solidsourceit.com

Web: www.solidsourceit.com